The Design, Development, and Implementation of a Web-based Control Data Tracking and Analysis System for Immunoassays Using Java Servlet Framework and Batch Oriented SAS Processing

How Tsao, John Sims, Andrew McKeen, Vivek Balakrishna, Khushroo Shroff
Vaccines R&D, WW Research and Development, Pfizer, 401 N. Middletown Road, Pearl River, NY 10965

Abstract

The Pfizer Vaccine Research department developed and implemented a new system architecture for the existing Control Data Tracking and Analysis System (CDTAS) [1], which provides tools for surveillance of various qualified bioanalytical assays. The primary goal of this project was to leverage SAS' traditional analytical and output capabilities in an interactive web application with non-proprietary software Integration Technology. The legacy system's SAS/AF® interface was replaced with a web interface developed using open source technology. SAS® components were redesigned to work in a batch-oriented environment, and without the need to run SAS as a local, client-side process. The web-based version of CDTAS was designed to work seamlessly with infrastructure components distributed across different sites and without the need for additional software licenses. The redesigned CDTAS system was successfully implemented in September of 2011.

Key Words: Control Data Tracking and Analysis, Statistical Process Control, Web Interface Application, Bioanalytical Assay, Batch Oriented SAS Programming, SAS

1. Introduction

The Pfizer Vaccine Research East and Early Development (VREED) group is responsible for immunoassay development, qualification, and high throughput testing to support vaccine preclinical studies and clinical trials. CDTAS is deployed in a controlled and regulated computer environment to provide tools for personnel to track the performance of quality control (QC) samples; the user can select one or more QC samples over any specified period of time and analyze the data in real time in several different, useful ways and view or report the data in ways that help understand how the process is performing day to day, or over any selected time period. It is most important for the performance monitoring of assays that require longevity - consistently reproducible and stable over months and years - such as vaccine projects that could go on for many years. The intended users for this application include business managers and laboratory personnel who are responsible for monitoring and improving assay operations. The user interface for the first release of CDTAS was developed using SAS/AF[1].

This paper describes key methods and lessons learned during the design and development of this application. An overview of the CDTAS system is given along with details of the design used by the project team to create a responsive and robust end-user web interface.

2. System Description

The first implementation of CDTAS was deployed to the end user as a fat-client application where desktop configuration is tightly controlled and file-system privileges are required to run SAS locally. The redesigned web-based application is deployed by granting users access to the URL through Internet Explorer (or any compatible web browser). The web-enabled version of CDTAS provides a single access point to several different statistical analysis modules. This allows one deployment of SAS on the server rather than many deployments across many PCs. SAS is used to generate all statistical calculations and to provide outputs in both PDF and HTML formats. SAS processing is carried out on the server side rather than on the user's computer. All data are stored in an Oracle database and accessed by SAS using SAS/ACCESS® to Oracle. The web component uses Oracle's JDBC driver to access data stored in Oracle.

CDTAS now provides:

- Web-based access by users from any Pfizer location
- Configuration of components that is consistent with Cloud Computing Environment – where servers and infrastructure components are offered as on-demand and driven by user requests from a remote PC or laptops
- Visual representations of the quality control sample results including Scatter Plots, Exponentially Weighted Moving Average (EWMA) Charts, X-Bar Charts, R Charts, Histograms, Line Plots, and Box Plots
- The ability to interactively drill down into detailed attributes of data points on the Scatter Plot or data table
- An output feature that will export data to Microsoft Excel enabling off-line analysis of data
- The ability to sort and dynamically filter data in the data table display using an interactive Search/Filter
- An interface with characteristics similar to those found in desktop applications
- A summary table of alerts triggered by a pre-defined set of testing rules – alerts the lab when there is nonrandom patterns in data

3. System Development

Overall, there were four primary challenges that the development team had to address:

- Provide a responsive web interface within the limits of the current IT infrastructure
- Provide integrated design of disparate technical infrastructure components
- Provide enhanced functionality to address the changing needs of the end user
- Provide effective management of system performance

3.1 **Provide a Responsive Web Interface with Available SAS Products**

The development team created a robust and dynamic web SAS application solution that provided a responsive web interface while leveraging the analytical capabilities of SAS. The team strived to provide interface characteristics similar to a desktop application. Specifically, interaction needed to be dynamic and screen updates made without redrawing the entire application screen. An application with characteristics similar to a desktop application is referred to as a Rich Internet Applications or RIA.

3.2 Provide Integrated Design of Disparate Technical Infrastructure Components

Infrastructure components of the CDTAS application reside at three different physical sites connected over a wide area network (WAN) as shown in Figure 1. Interaction among components needed to be designed so that network traffic between the following components was minimized:

- Database Tier: Oracle DBMS in Site A
- Web Tier: JBoss Web Server in Site B
- SAS Tier: SAS System Installation in Site C

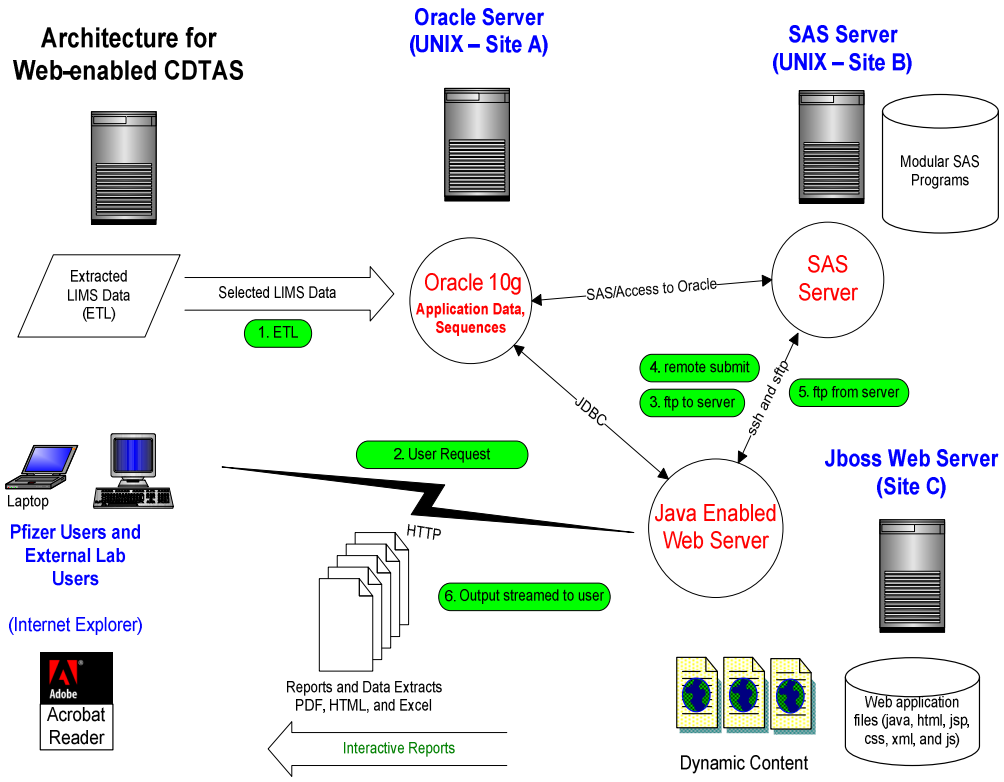


Figure 1: Web-enabled CDTAS Infrastructure Diagram

SAS Tier

The SAS components were redesigned to work in a batch-oriented environment in order to work seamlessly with the open-source JBOSS web server platform located at Site C. The processing overhead of invoking and terminating SAS was incurred for each report run on the SAS Server located at Site B. Administration of the SAS components is isolated on the SAS Server where a single installation of SAS is used. Securing and administrating a single installation of SAS is much easier than maintaining multiple installations on all PCs used to run the SAS/AF version of CDTAS. The following SAS components were used in the web-enabled version:

- BASE SAS®
- SAS/ACCESS®
- SAS/STAT®
- SAS/QC®
- SAS/GRAPH®
- SAS Output Delivery System (ODS)

The SAS tier was implemented as a series of macros and UNIX Shell scripts. The development team created a well-defined specification for passing control to and from the SAS tier from the web tier. Having a clearly defined specification facilitates division of duties by allowing the SAS programmers and statisticians to focus on data analysis and

output presentation using SAS. SAS programmers and statisticians did not need to know Java or the web tier configuration. The web developers provided the interface for filtering the data and selecting the appropriate output parameters. The web tier transfers control to the SAS tier using macro variables. The SAS tier generates the output which is streamed to the browser once SAS processing is complete.

The team transformed the old PC SAS framework of version 9.1.3 to current Pfizer's SAS version 9.2 on a SAS server. To make it work, conversions had to be made in ODS PDF and ODS LAYOUT which enabled successful generation of SAS output through the web. Technical issues were successfully addressed using online support and SAS authored papers [2]. Hardware and software font differences, as well as changes to the fonts used in SAS version 9.1.3 were also successfully converted.

In order to effectively manage the entire life cycle of an application at Pfizer, four areas must be maintained including development, test, stage, and production for all tiers. Each tier required its own area for each of the steps. The SAS Server tier facilitated this using a directory structure separating each area. Appendix A gives a description of the SAS Server structure. The directory structure enables the specification created to uniquely identify each report request. For example, a directory is created in the output directory that's equivalent to the run id generated from the Oracle sequence.

Web Tier

Business logic implemented in the Screen Control Language (SCL) for the SAS/AF version of CDTAS was reworked and coded in the web-tier using Java to support the web-based version of CDTAS. The web tier (or middle tier) was implemented using the Java Servlet specification. Interaction with users was implemented using several open source technologies used to develop Rich Internet Applications (RIA). JQuery was extensively used as a wrapper to interface components using Asynchronous JavaScript and XML (AJAX).

The following software components and scripting frameworks were used to develop the web-tier:

- Java 6 including various Libraries listed below.
    - j2ssh – remote execution of Shell script to invoke SAS; sftp of entry point program to SAS server and sftp of output (PDF and RTF) to client
    - ojdbc14 – JDBC driver
    - LDAP – Authentication is done using LDAP
    - poi 2.5.1 - Java API for Microsoft Documents that enables creation of Excel spreadsheet
- JBoss 5.0.1 or later – Web/Application Server
- HTML/CSS/JavaScript
- JQuery JavaScript library along with the following plugins to implement RIA characteristics:
    - AnyTime (http://www.ama3.com/anytime)
    - ThickBox (http://jquery.com/demo/thickbox)
    - DataTables (http://www.datatables.net)

Microsoft Windows Internet Explorer Version 7 or higher – Client

Database Tier

The database tier was implemented using Oracle 10g. Database objects included tables, views, and sequences. The tables and views are accessed by the middle tier using an Oracle account with read-only privileges to the tables. The connections are managed using JBoss' connection pool facility.

The following explanation provides an overview of application flow. The primary steps in a report request are highlighted in Figure 1 above using green rectangles with rounded edges:

1. ETL : Application Data is provided using an Extract, Transform, Load (ETL) process that is separate from the web interface. Data is extracted from the data collection system (LIMS) and stored in a schema accessed by CDTAS.
2. User Request: Users navigate selections to select filter criteria (Lab, Study, Platform, Test Group, Serotype(s), Quality Sample Name(s) and time frame). The dynamic or interactive HTML for the web interface is implemented using JavaScript and CSS. Report parameters are selected and the report request submitted.
3. ftp to server: A batch SAS program is created and transferred to the SAS server using secure ftp (sftp). An integer is generated for each request from an Oracle sequence. The unique integer is used by the middle tier and SAS tier to name temporary files used for each request. Guaranteeing files are uniquely named across requests insures there are no input and output filename collisions for different requests. The SAS log file is also unique to each run by embedding the run number in the output log file name.
4. Remote Submit: A secure session is established from the web server to the SAS server. The web application then calls a UNIX Shell script on the SAS server to run the appropriate SAS program. The parameter to the shell script is the unique run number generated from the Oracle Sequence. The run number identifies the program entry point sftp'ed in step 2 above. The SAS batch program (report request entry point) uses macro variables to pass criteria needed to generate the results.
5. ftp from server: The output (PDF or HTML and corresponding Image) are ftp'ed to the web server once the SAS batch program finishes. The output file was transferred using sftp from the SAS server to the web server where it could be displayed in an <iframe>. Using an <iframe> in the main page was necessary because the users did not want multiple output windows popping up.
6. Output displayed to user: The output area's (iframe's 'src' attribute) is redirected to the PDF or HTML output.

3.3. Provide Enhanced Functionality

Additional functionality was developed to improve the user experience by providing more data-specific information and facilitate user selection of a valid date range.

Scatter plots with drill through capability

SAS/AF allows multiple views to be attached to the same class representing the data (model). The SAS/AF version of CDTAS had a table view and graph view attached to the same model. This allowed the user to select points on the graph and have them automatically highlighted in the table. The development team for the web-based version implemented a "drill-thru" capability using the HTML output feature of the Output Delivery System (ODS). The Scatter Plot – Log HTML report is an HTML report with an area map for the graphical image used to implement drill-thru. The area map defines points and hyperlinks used to drill through to the detail data. Clicking on a point in the graph pops up the informational dialog as shown in Figure 2. A post-processing step was needed to augment the HTML output of ODS that built a URL for each of the points on a graph. The URL pointed to a Java Servlet that queried detail data based on the point identifier (primary key).

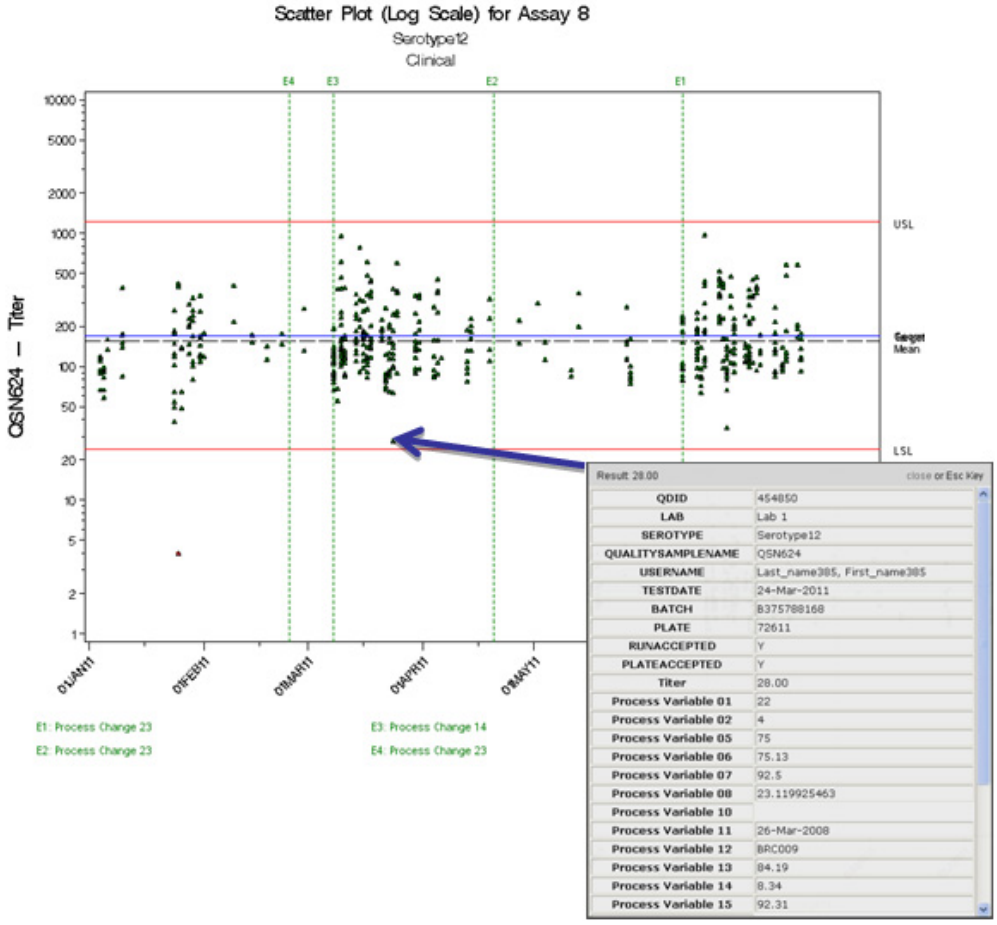
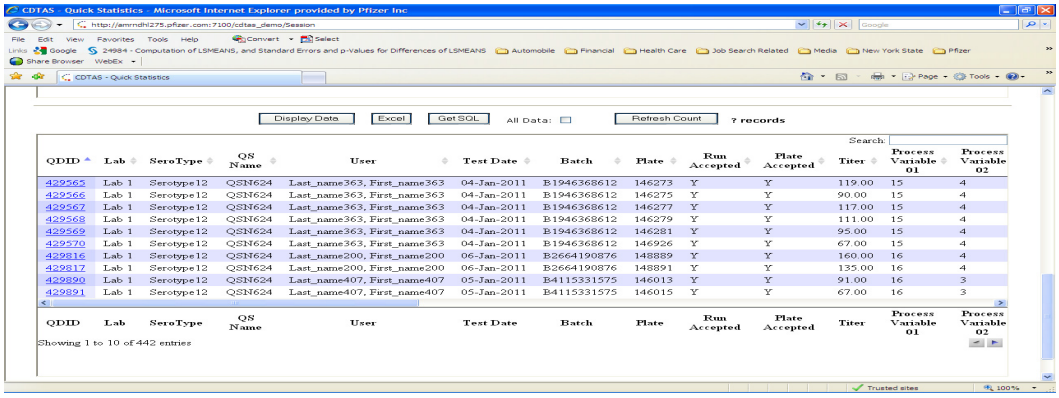


Figure 2: Scatter Plot – Log HTML with Informational Dialog Popup Window

The popup window contains the result value and "(OOL)" appended to the result value if the value/point is out of limits. The JQuery ThickBox plug-in (http://jquery.com/demo/thickbox) was used to implement the pop-up. A code snippet illustrating how SAS' ODS HTML output is augmented to provide drill-thru capability can be found in Appendix B.

Data table display with dynamic filtering, sorting, and drill through

Users needed the ability to easily understand the profile of data meeting the filter criteria. The interface needed to be intuitive and dynamic. The development team implemented an interactive data table (Display 1) to accomplish this requirement. The data table allowed dynamic searching and sorting of data meeting filter criteria. The data table was implemented using the JQuery DataTables plug-in (http://www.datatables.net). Data was queried using a Java Servlet and returned using JavaScript Object Notation (JSON).

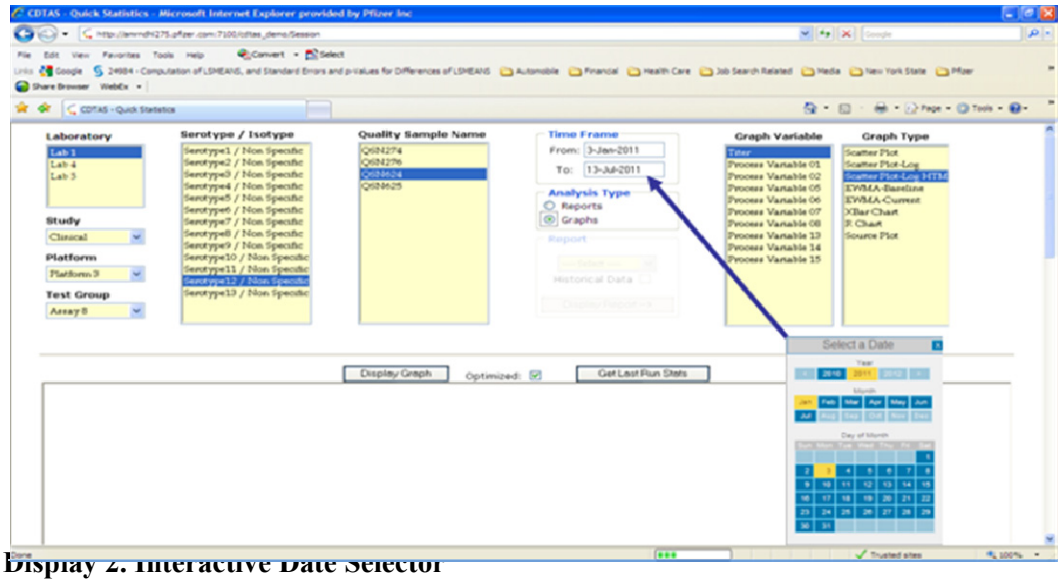


Display 1. Interactive Data Table

The data table also indicates the number of observations/entries meeting the filter criteria. The number of entries gives the user an indication of how long graph generation would take.

Automatically deployed Date selector on time frame field

Users needed the ability to constrain data to a specific date range. The interface needed to limit the user's date range to valid dates. That is, only dates containing data meeting the current filter criteria should be allowed. A JQuery plug-in was used to implement the Time Frame filter which is based on the attribute test date in one of the Oracle tables. The *from* and *to* fields are initially filled in with the minimum and maximum test dates for the selected criteria (lab, study, platform, etc…). Changes to the dates can be made by clicking in the field. An interactive date selector pops up and allows the user to select a date (see Display 2). The date selector has been configured to allow the user to only select a date containing data (i.e. dates outside the minimum and maximum dates are not selectable):



Display 2. Interactive Date Selector

The date selector is implemented using the jQuery anyTime plug-in:
http://www.ama3.com/anytime

3.4. Provide Effective Management of System Performance

Pfizer consolidated SAS licenses and the SAS infrastructure is shared across many applications, both interactive and batch. The system load was very sporadic and hard to predict. Developing an interactive system with acceptable performance was challenging but it does allow flexibility with respect to a dynamic organization where people and infrastructure are often located at different sites. There were several points of potential contention as illustrated in Figure 1. For example, all of the servers have a variable load throughout the day which may noticeably impact the time it takes to display a graph. Graphs with one specific quality sample could run in less than fifteen seconds but could take longer depending on server and network load. Several iterations of performance assessment and tuning occurred. The following describes measures the team implemented to manage system performance.

Create Database Indexes

Users needed a cascading effect while selecting filter criteria. Features not available for the current intersection of attribute values selected should also be made unavailable. The cascading filter and surfacing of application features was accomplished using Asynchronous JavaScript and XML (AJAX). Using the JQuery library simplified the AJAX calls used to provide cascading data selection. The database was optimized to return query results used to populate selections in a timely manner. Selection fields were implemented as cascading selections. For example, users would choose a lab and the list of available platforms would be populated. Indexes were created on the appropriate tables so the Servlet returning selection criteria executed queries quickly. These indexes also reduced the amount of time it took to generate a report.

Leverage SQL Pass-through

Dataset volume and location were not an issue in the SAS/AF version. The datasets were small enough that local copies were made and formats created during the application

initialization. These local copies were used for report requests so the performance of screen interaction and report generation were fast and predictable.

Since the SAS Server and Oracle Server were at different sites and connected over a WAN in the web-enabled version of CDTAS, minimizing the volume of data traveling over the network between SAS and Oracle was critical. This was accomplished using SQL Pass-through for queries involving large datasets. Some of the transformations that existed in SAS for the SAS/AF version needed to be implemented in the Query that was passed through to Oracle. Leveraging SAS' Oracle Pass-through returned the largest performance gain.

Increase Operating System Priority of SAS using the UNIX *nice* command

Giving the SAS processes a higher operating system priority (UNIX nice value) resulted in more consistent run times between runs. The SAS server is used for SAS batch processing unrelated to CDTAS. The default nice value for all SAS processes was 10. Lowering the nice value to 0 for CDTAS programs decreased run times and improved predictability.

Indicate number of entries/records to be processed

The SAS programs are computationally intensive and reporting on large datasets can take considerable time. The interface provides a mechanism for the user to ascertain how many entries/records were going to be processed. This is a useful feature to help assess response time when working with large datasets. The number of entries meeting the current filter criteria was reported in the data table display (Display 2 above).

The number of entries also matched the number of observations/records you would see in an exported Excel spreadsheet and was a good indicator of the time a report request would take. For example, a report processing 500 records would run much faster than a report run processing 10,000 records.

Use Cross-over design of experiment to demonstrate reduction in system response time

To demonstrate results of the performance improvement, the team conducted a small experiment comparing run times before and after optimization. For a selected control chart, results of run times before optimization and after optimization were recorded repeatedly in small time blocks throughout the day (See Figure 3). Red X symbols represent run times before optimization and blue circles represent run times after optimization. Note that run time performance after optimization is more consistent and predictable than run time performance before optimization.

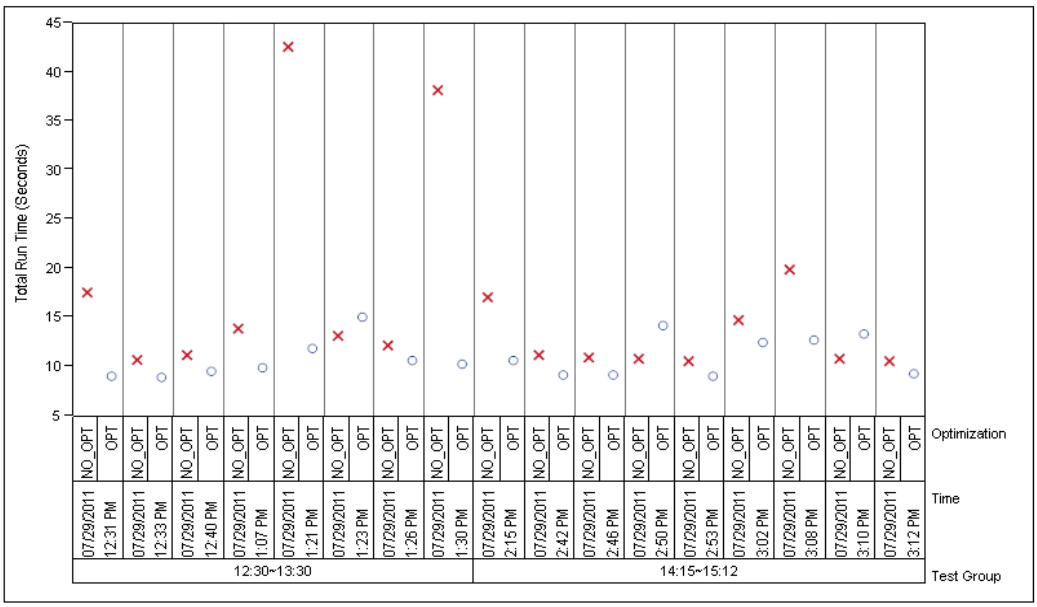


Figure 3. Optimization Results

Leverage the Cloud computing Strategy

Cloud computing is a recent buzzword being used in the industry. The National Institute of Standards and Technology (NIST) provides a concise and specific definition:

> Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Cloud computing provides computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Pfizer is exploring the use of internal and external clouds. The web-enabled version of CDTAS is designed to leverage the Cloud Computing model once it is mature within Pfizer.

Path Forward

The application team will continue to look for ways to improve performance. A framework that allows for a persistent remote SAS process would eliminate the overhead of starting and stopping SAS for each report request. The SAS/EBI platform may be licensed in the future.

Acknowledgements

The authors would like to thank the following individuals who have made the new release of CDTAS possible: Jay Billy, Chris Gosselin, Kathrin Jansen, Mike Longden, Ray Borzone, and Jeff Stroosnyder.

References

[1] How Tsao, et. al. (2008), "The Design, Development and Implementation Of A Control Data Tracking And Analysis System For Immunoassays Using SAS/AF", Proceedings of the SAS Global Forum 2008 Conference. Available at http://www2.sas.com/proceedings/forum2008/020-2008.pdf

[2] Huntley, S., Lawhorn, B. (2010), "Getting the Right Report (Again): Your Compatibility Guide for ODS PDF 9.2", Proceedings of the SAS Global Forum 2010 Conference. Available at http:// support.sas.com/resources/papers/proceedings10/035-2010.pdf

SAS® Software 9.2 (TS2M3) Running on SunOS 5.9 (SUN 64) Platform
SAS Institute Inc. Cary, NC, USA

Documentation for SAS® Products and Solutions:
http://support.sas.com/documentation/92/index.html

- SAS/ACCESS® 9.2 for Relational Databases: Reference, Fourth Edition
- SAS/GRAPH® 9.2 Reference, Second Edition
- SAS/QC® 9.2: User's Guide, Second Edition
    - Chapter 9. The MACONTROL Procedure
    - Chapter 13. The SHEWHART Procedure
- SAS/STAT® 9.2 User's Guide, Second Edition
    - Chapter 24. The BOXPLOT Procedure
- SAS® 9.2 Macro Language: Reference
- SAS® 9.2 Output Delivery System: User's Guide
    - Part 5. The Template Procedure
- Base SAS® 9.2 Procedures Guide
    - Chapter 55 – The SQL Procedure

Appendix A. SAS Server Directory Structure

CDTAS	Top level directory.
|---Environment (devel, test, stage, or prod)	SDLC environment.
|---bin	Directory for shell scripts.
|---control | |	SAS autoexec file and crontab files for the different CDTAS UNIX accounts (cdtasxd, cdtasxt, and cdtasxp)
| ---log | | | |	Summary log file for all activity (report runs and cleanup runs); Individual SAS run log files are saved here if SAS ends with non-zero return code. SAS ends with a return code of 1 if there are warnings in the program. A return code of 2 indicates an error.
|---output | |	Top level directory where report output (PDF and HTML) is stored
| |---<run id> | | | |	Run id is generated from an Oracle sequence and unique for each report run; html output is stored in a subdirectory that's equal to the run id
|---queue | | |	A run-specific SAS source file is created for each report request; The queue directory contains the SAS source file with the unique run id embedded in the name
|---sas |	Directory containing the SAS programs; Design of the SAS programs is in a separate document
|---tmp | | |	Each report run creates several temporary files (for example, the sasuser directory and graph output); A unique directory is equivalent to the run id and will be used by the report run

APPENDIX B. CODE SNIPPET

The following code was generated in a Java Servlet: and contains the following items:

- styles/thickbox.css – style sheet for thickbox plug-in
- <map name="g5adgaak_map"> - image map created by ODS HTML
- <area> tags created for each point on the plot; The ODS HTML had several limitations and this section was augmented by an intermediate Servlet; The 'href="Session…"' references the Session Servlet that's used to retrieve results for point identified by 413080
- img src="/cdtas_tmp/4877/adp.png" – ping image generated by ODS HTML
- <script> tags – inclusion of base jQuery plug-in and thickbox plug-in

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="styles/thickbox.css" />
  </head>

  <body>
    <div align='center'>
      <map name="g5adgaak_map">
        <area shape="rect" alt="Result: 12.29" title="Result: 12.29"
href="Session?cmd=display-
qdid&id=413080&TB_iframe=true&height=500&width=450" class="thickbox nool"
id="413080" coords="856,254,859,257"/>
<area shape="rect"  alt="Result: 14.04" title="Result: 14.04"
href="Session?cmd=display-
qdid&id=413071&TB_iframe=true&height=500&width=450" class="thickbox nool"
id="413071" coords="856,244,859,247"/>
… etc (area for each point on the plot)…
      </map>
      <img src="/cdtas_tmp/4877/adp.png" usemap="#g5adgaak_map" />
    </div>
    <script type="text/javascript" language="javascript" src="js/jquery-
1.4.2.js"></script>
    <script type="text/javascript" language="javascript"
src="js/thickbox.js"></script>
  </body>
</html>
```